

Münchausen-számok keresése (alap)

Kovács Csaba, UBMCGU

Szkriptnyelvek (INBPM9942L) 2021. tavasz

Laborvezető: Dr. Szathmáry László

Áttekintés

- Münchausen-szám keresés
- Naiv módon
- Direkt a gagyi `str(num)` felbontás, semmi % base mágia
- Először ciklussal
- Utána list comprehension-nel átírjuk
- Repo: <https://github.com/rkeeves/munchausen>

Münchausen-szám definíció

- Egy természetes számot akkor nevezünk Münchausen-számnak, ha minden egyes számjegyet az adott számjegy által meghatározott hatványra emelve, majd ezen hatványok összegét véve az eredeti számot kapjuk vissza.
- Most legyen $0^0 = 0$
- Sok a szöveg... nézzünk példát!

Münchausen-szám definíció

- Egy természetes számot akkor nevezünk Münchausen-számnak, ha minden egyes számjegyet az adott számjegy által meghatározott hatványra emelve, majd ezen hatványok összegét véve az eredeti számot kapjuk vissza.
- Most legyen $0^0 = 0$
- Példa 2 0 2 1

Münchausen-szám definíció

- Egy természetes számot akkor nevezünk Münchausen-számnak, ha minden egyes számjegyét az adott számjegy által meghatározott hatványra emelve, majd ezen hatványok összegét véve az eredeti számot kapjuk vissza.
- Most legyen $0^0 = 0$

- Példa

2	0	2	1
2^2	0^0	2^2	1^1

Münchausen-szám definíció

- Egy természetes számot akkor nevezünk Münchausen-számnak, ha minden egyes számjegyét az adott számjegy által meghatározott hatványra emelve, majd ezen hatványok összegét véve az eredeti számot kapjuk vissza.
- Most legyen $0^0 = 0$

- Példa

2	0	2	1
2^2	0^0	2^2	1^1
4	0	4	1

Münchausen-szám definíció

- Egy természetes számot akkor nevezünk Münchausen-számnak, ha minden egyes számjegyét az adott számjegy által meghatározott hatványra emelve, majd ezen hatványok összegét véve az eredeti számot kapjuk vissza.
- Most legyen $0^0 = 0$

- Példa

$$\begin{array}{cccc} 2 & 0 & 2 & 1 \\ 2^2 & 0^0 & 2^2 & 1^1 \\ 4+ & 0+ & 4+ & 1 = 9 \end{array}$$

Münchausen-szám definíció

- Egy természetes számot akkor nevezünk Münchausen-számnak, ha minden egyes számjegyét az adott számjegy által meghatározott hatványra emelve, majd ezen hatványok összegét véve az eredeti számot kapjuk vissza.
- Most legyen $0^0 = 0$

- Példa

$$\begin{array}{cccc} 2 & 0 & 2 & 1 \\ 2^2 & 0^0 & 2^2 & 1^1 \end{array}$$

$$4 + 0 + 4 + 1 = 9 \quad ? = 2021$$

Münchausen-szám definíció

- Egy természetes számot akkor nevezünk Münchausen-számnak, ha minden egyes számjegyét az adott számjegy által meghatározott hatványra emelve, majd ezen hatványok összegét véve az eredeti számot kapjuk vissza.
- Most legyen $0^0 = 0$

- Példa

$$\begin{array}{cccc} 2 & 0 & 2 & 1 \\ 2^2 & 0^0 & 2^2 & 1^1 \end{array}$$

$$4 + 0 + 4 + 1 = 9 \neq 2021$$

Münchausen-szám definíció

- Egy természetes számot akkor nevezünk Münchausen-számnak, ha minden egyes számjegyét az adott számjegy által meghatározott hatványra emelve, majd ezen hatványok összegét véve az eredeti számot kapjuk vissza.
- Most legyen $0^0 = 0$

- Példa

$$\begin{array}{cccc} 2 & 0 & 2 & 1 \\ 2^2 & 0^0 & 2^2 & 1^1 \end{array}$$

$$4 + 0 + 4 + 1 = 9 \neq 2021$$

Nem Münchausen-szám mert \neq

Münchausen-szám definíció

- Nézzük meg most 3435-öt!

Münchausen-szám definíció

- Nézzük meg most 3435-öt!

3 4 3 5

Münchausen-szám definíció

- Nézzük meg most 3435-öt!

$$\begin{array}{cccc} 3 & 4 & 3 & 5 \\ 3^3 & 4^4 & 3^3 & 5^5 \end{array}$$

Münchausen-szám definíció

- Nézzük meg most 3435-öt!

$$\begin{array}{cccc} 3 & 4 & 3 & 5 \\ 3^3 & 4^4 & 3^3 & 5^5 \\ 27 & 256 & 27 & 3125 \end{array}$$

Münchausen-szám definíció

- Nézzük meg most 3435-öt!

3 4 3 5

3^3 4^4 3^3 5^5

27+ 256+ 27+ 3125

Münchausen-szám definíció

- Nézzük meg most 3435-öt!

3 4 3 5

3^3 4^4 3^3 5^5

$27 + 256 + 27 + 3125 = 3435$

Münchausen-szám definíció

- Nézzük meg most 3435-öt!

3 4 3 5

3^3 4^4 3^3 5^5

$27 + 256 + 27 + 3125 = 3435$ **?** $= 3435$

Münchausen-szám definíció

- Nézzük meg most 3435-öt!

3 4 3 5

3^3 4^4 3^3 5^5

$27 + 256 + 27 + 3125 = 3435 == 3435$

Münchausen-szám definíció

- Nézzük meg most 3435-öt!

3 4 3 5

3^3 4^4 3^3 5^5

$27 + 256 + 27 + 3125 = 3435 == 3435$

Münchausen-szám mert $==$

Münchausen-szám definíció

- Algoritmizálható-e amit eddig kézzel csináltunk?

Münchausen-szám definíció

- Algoritmizálható-e amit eddig kézzel csináltunk?
- Nézzük meg naivan ciklussal Python3-ban!

Münchausen-szám teszt (naiv, ciklus)

- Vegyünk egy számot „num”

```
def is_munchausen(num):
```

Münchausen-szám teszt (naiv, ciklus)

- Átkonvertáljuk string-é, és végig megyünk a számjegy karaktereken

```
def is_munchausen(num):  
    for digit_char in str(num):
```

Münchausen-szám teszt (naiv, ciklus)

- A számjegy karaktereket vissza kell konvertálni számmá
- Mert aritmetikai műveletet ugye nem tudnánk string-en végezni :(

```
def is_munchausen(num):  
    for digit_char in str(num):  
        digit = int(digit_char)
```


Münchausen-szám teszt (naiv, ciklus)

- Simán önmaga hatványára emeljük

```
def is_munchausen(num):  
    for digit_char in str(num):  
        digit = int(digit_char)  
        digit ** digit
```

Münchausen-szám teszt (naiv, ciklus)

- Simán önmaga hatványára emeljük $3^3 + 4^4 + 3^3 + 5^5 = 3435 == 3435$
- Na jó... de ezt valahogyan szummázni is kéne

```
def is_munchausen(num):  
    for digit_char in str(num):  
        digit = int(digit_char)  
        digit ** digit
```

Münchausen-szám teszt (naiv, ciklus)

- Szummázzuk egy változóba!

```
def is_munchausen(num):  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit
```

Münchausen-szám teszt (naiv, ciklus)

- Inicializáljuk is azért

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit
```

Münchausen-szám teszt (naiv, ciklus)

- Nézzük meg, hogy egyenlő-e a számmal

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit  
    return accu == num
```

Münchausen-szám teszt (naiv, ciklus)

- Nézzük meg, hogy egyenlő-e a számmal
- Példa pls

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit  
    return accu == num
```

Münchausen-szám teszt (naiv, ciklus)

- Nézzük meg, hogy egyenlő-e a számmal
- Példa pls

3435 == 3435

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit  
    return accu == num
```

Münchausen-szám teszt (naiv, ciklus)

- Nézzük meg, hogy egyenlő-e a számmal
- Példa pls

3435 == 3435
9 != 2021

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit  
    return accu == num
```


Münchausen-szám teszt (naiv, ciklus)

- Kész! Azért próbáljuk ki hátha buggy!

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit  
    return accu == num
```

Münchausen-szám teszt (naiv, ciklus)

- Teszteljük! (link:
https://github.com/rkeeves/munchausen/blob/main/src/e00_munchausen_property.py
)

Münchausen-szám teszt (naiv, ciklus)

- Teszteljük!

```
0 -> False  
1 -> True  
2021 -> False  
3435 -> True
```

Münchausen-szám teszt (naiv, ciklus)

- Teszteljük!
- Hopp, 0-nál eltérés van :(

```
0 -> False  
1 -> True  
2021 -> False  
3435 -> True
```

Münchausen-szám teszt (naiv, ciklus)

- Teszteljük!
- Hopp, 0-nál eltérés van :(
- Miért is? $\rightarrow 0^{**} 0 = 1$

```
0 -> False
1 -> True
2021 -> False
3435 -> True
```

Münchausen-szám teszt (naiv, ciklus)

- Teszteljük!
- Hopp, 0-nál eltérés van :(
- Miért is? $\rightarrow 0 ** 0 = 1$
- „Javítsuk gyorsan és felületesen”

Münchausen-szám teszt (naiv, ciklus)

- Teszteljük!
- Hopp, 0-nál eltérés van :(
- Miért is? $\rightarrow 0 ** 0 = 1$
- „Javítsuk gyorsan és felületesen”

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += 0 if digit == 0 else digit ** digit  
    return accu == num
```

Münchausen-szám teszt (naiv, ciklus)

- Teszteljük!
- Hopp, 0-nál eltérés van :(
- Miért is? $\rightarrow 0 ** 0 = 1$
- Teszteljük! (link:
https://github.com/rkeeves/munchausen/blob/main/src/e01_munchausen_property.py
)

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += 0 if digit == 0 else digit ** digit  
    return accu == num
```


Münchausen-szám teszt (naiv, ciklus)

- Teszteljük!
- Hopp, 0-nál eltérés van :(
- Miért is? $0 ** 0 = 1$
- Teszteljük!

```
0 -> True  
1 -> True  
2021 -> False  
3435 -> True
```

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += 0 if digit == 0 else digit ** digit  
    return accu == num
```

Így már OK

Münchausen-szám listázás (naiv, ciklus)

- Már el tudjuk dönteni hogy mi eleme a Münchausen-számok halmazának!

Münchausen-szám listázás (naiv, ciklus)

- Már el tudjuk dönteni hogy mi eleme a Münchausen-számok halmazának!
- De hogyan listázzuk ki?

Münchausen-szám listázás (naiv, ciklus)

- Már el tudjuk dönteni hogy mi eleme a Münchausen-számok halmazának!
- De hogyan listázzuk ki?
- Terv: Soroljuk fel a számokat, és válogassuk ki a Münchause -számokat

Münchausen-szám listázás (naiv, ciklus)

- Már el tudjuk dönteni hogy mi eleme a Münchausen-számok halmazának!
- De hogyan listázzuk ki?
- Terv: Soroljuk fel a számokat, és válogassuk ki a Münchausen-számokat
- Nézzük meg algoritmizálva Python3-ban, naivan, ciklussal

Münchausen-szám listázás (naiv, ciklus)

- Vegyünk egy limitet „up_to” néven (pl. 10000)

```
def list_munchausen(up_to):
```

Münchausen-szám listázás (naiv, ciklus)

- Menjünk végig az összes számon ezen az intervallumon $[0, \text{up_to}[$

```
def list_munchausen(up_to):  
    for num in range(0, up_to):
```

Münchausen-szám listázás (naiv, ciklus)

- Ha valami Münchausen-szám, csak akkor csinálunk valamit
- De mit is csináljunk?????
- User-nek egy listát kell visszaadni a Münchausen-számokkal

```
def list_munchausen(up_to):  
    for num in range(0, up_to):  
        if is_munchausen(num):
```


Münchausen-szám listázás (naiv, ciklus)

- Nos, akkor dobjuk bele egy listába...

```
def list_munchausen(up_to):  
    for num in range(0, up_to):  
        if is_munchausen(num):  
            accu.append(num)
```

Münchausen-szám listázás (naiv, ciklus)

- Inicializáljuk is, mielőtt könnyre fakad

```
def list_munchausen(up_to):  
    accu = []  
    for num in range(0, up_to):  
        if is_munchausen(num):  
            accu.append(num)
```

Münchausen-szám listázás (naiv, ciklus)

- Adjuk vissza a listát!

```
def list_munchausen(up_to):  
    accu = []  
    for num in range(0, up_to):  
        if is_munchausen(num):  
            accu.append(num)  
    return accu
```

Münchausen-szám listázás (naiv, ciklus)

- Kész! Azért próbáljuk ki hátha buggy!

```
def list_munchausen(up_to):  
    accu = []  
    for num in range(0, up_to):  
        if is_munchausen(num):  
            accu.append(num)  
    return accu
```

Münchausen-szám listázás (naiv, ciklus)

- Teszteljük! (link: https://github.com/rkeeves/munchausen/blob/main/src/e02_munchausen_property.py)

```
def list_munchausen(up_to):  
    accu = []  
    for num in range(0, up_to):  
        if is_munchausen(num):  
            accu.append(num)  
    return accu
```

Münchausen-szám listázás (naiv, ciklus)

- Teszteljük!

```
1 -> [0]
10 -> [0, 1]
100 -> [0, 1]
1000 -> [0, 1]
10000 -> [0, 1, 3435]
100000 -> [0, 1, 3435]
1000000 -> [0, 1, 3435]
```

```
def list_munchausen(up_to):
    accu = []
    for num in range(0, up_to):
        if is_munchausen(num):
            accu.append(num)
    return accu
```

Münchausen-szám listázás (naiv, ciklus)

- Teszteljük!
- Úgy látszik működik...

```
1 -> [0]
10 -> [0, 1]
100 -> [0, 1]
1000 -> [0, 1]
10000 -> [0, 1, 3435]
100000 -> [0, 1, 3435]
1000000 -> [0, 1, 3435]
```

```
def list_munchausen(up_to):
    accu = []
    for num in range(0, up_to):
        if is_munchausen(num):
            accu.append(num)
    return accu
```

Münchausen-szám listázás (naiv, ciklus)

- De azért nézzünk már rá a kódra...

Münchausen-szám listázás (naiv, ciklus)

- De azért nézzünk már rá a kódra...
- Túl sok a boiler plate.

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += 0 if digit == 0 else digit ** digit  
    return accu == num  
  
def list_munchausen(up_to):  
    accu = []  
    for num in range(0, up_to):  
        if is_munchausen(num):  
            accu.append(num)  
    return accu
```

Münchausen-szám listázás (naiv, ciklus)

- De azért nézzünk már rá a kódra...
- Túl sok a boiler plate.
- Próbáljuk meg tömörebben kifejezni magunkat!

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += 0 if digit == 0 else digit ** digit  
    return accu == num  
  
def list_munchausen(up_to):  
    accu = []  
    for num in range(0, up_to):  
        if is_munchausen(num):  
            accu.append(num)  
    return accu
```

Münchausen-szám listázás (naiv, list comprehension)

- List comprehension-t használjuk

Münchausen-szám listázás (naiv, list comprehension)

- List comprehension-t használjuk
- Balra kirakom a régit „emlékeztetőül”

Münchausen-szám listázás (naiv, list comprehension)

- Először a `list_munchausen`-t!

Münchausen-szám listázás (naiv, list comprehension)

- Először a list_munchausen-t!

```
def list_munchausen(up_to):  
    accu = []  
    for num in range(0, up_to):  
        if is_munchausen(num):  
            accu.append(num)  
    return accu
```

Münchausen-szám listázás (naiv, list comprehension)

- Vegyünk egy limitet „up_to” néven (pl. 10000)

```
def list_munchausen(up_to):  
    accu = []  
    for num in range(0, up_to):  
        if is_munchausen(num):  
            accu.append(num)  
    return accu
```

```
def list_munchausen(up_to):  
    |
```

Münchausen-szám listázás (naiv, list comprehension)

- Menjünk végig az összes számon ezen az intervallumon [0,up_to[

```
def list_munchausen(up_to):  
    accu = []  
    for num in range(0, up_to):  
        if is_munchausen(num):  
            accu.append(num)  
    return accu
```

```
def list_munchausen(up_to):  
    num in range(0, up_to)
```


Münchausen-szám listázás (naiv, list comprehension)

- Ha valami Münchausen-szám, csak akkor érintjük az elemet (iterálásban)

```
def list_munchausen(up_to):  
    accu = []  
    for num in range(0, up_to):  
        if is_munchausen(num):  
            accu.append(num)  
    return accu
```

```
def list_munchausen(up_to):  
    [num for num in range(0, up_to) if is_munchausen(num)]
```

Münchausen-szám listázás (naiv, list comprehension)

- Adjuk vissza listaként!

```
def list_munchausen(up_to):  
    accu = []  
    for num in range(0, up_to):  
        if is_munchausen(num):  
            accu.append(num)  
    return accu
```

```
def list_munchausen(up_to):  
    return [num for num in range(0, up_to) if is_munchausen(num)]
```

Münchausen-szám listázás (naiv, list comprehension)

- Kész! Azért nézzük meg!

```
def list_munchausen(up_to):  
    accu = []  
    for num in range(0, up_to):  
        if is_munchausen(num):  
            accu.append(num)  
    return accu
```

```
def list_munchausen(up_to):  
    return [num for num in range(0, up_to) if is_munchausen(num)]
```

Münchausen-szám listázás (naiv, list comprehension)

- Teszteljük! (link: https://github.com/rkeeves/munchausen/blob/main/src/e03_munchausen_property.py)

```
def list_munchausen(up_to):  
    return [num for num in range(0, up_to) if is_munchausen(num)]
```

Münchausen-szám listázás (naiv, list comprehension)

- Teszteljük!

```
1 -> [0]
10 -> [0, 1]
100 -> [0, 1]
1000 -> [0, 1]
10000 -> [0, 1, 3435]
100000 -> [0, 1, 3435]
1000000 -> [0, 1, 3435]
```

```
def list_munchausen(up_to):
    return [num for num in range(0, up_to) if is_munchausen(num)]
```

Münchausen-szám listázás (naiv, list comprehension)

- Teszteljük!
- Jónak tűnik

```
1 -> [0]
10 -> [0, 1]
100 -> [0, 1]
1000 -> [0, 1]
10000 -> [0, 1, 3435]
100000 -> [0, 1, 3435]
1000000 -> [0, 1, 3435]
```

```
def list_munchausen(up_to):
    return [num for num in range(0, up_to) if is_munchausen(num)]
```

Münchausen-szám listázás (naiv, list comprehension)

- Teszteljük!
- Jónak tűnik
- Nézzük most az `is_munchausen`-t!

```
1 -> [0]
10 -> [0, 1]
100 -> [0, 1]
1000 -> [0, 1]
10000 -> [0, 1, 3435]
100000 -> [0, 1, 3435]
1000000 -> [0, 1, 3435]
```

```
def list_munchausen(up_to):
    return [num for num in range(0, up_to) if is_munchausen(num)]
```

Münchausen-szám teszt (naiv, list comprehension)

- Vagy iterátort, vagy list comprehension-t használjunk
- Balra kirakom a régit „emlékeztetőül”

Münchausen-szám teszt (naiv, list comprehension)

- Vegyünk egy számot „num”

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit  
    return accu == num
```

```
def is_munchausen(num):  
    |
```

Münchausen-szám teszt (naiv, list comprehension)

- Vissza kell adnunk hogy egyenlő-e a szumma a számmal

$$27 + 256 + 27 + 3125 = 3435 == 3435$$

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit  
    return accu == num
```

```
def is_munchausen(num):  
    return
```

`== num`

Münchausen-szám teszt (naiv, list comprehension)

- Vissza kell adnunk hogy egyenlő-e a szumma a számmal
- De hogy írom le a szummát python-ban?

27+ 256+ 27+ 3125 = 3435 == 3435

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit  
    return accu == num
```

```
def is_munchausen(num):  
    return
```

== num

Münchausen-szám teszt (naiv, list comprehension)

- Sum :)

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit  
    return accu == num
```

```
def is_munchausen(num):  
    return sum(  
        ) == num
```

Münchausen-szám teszt (naiv, list comprehension)

- Sum :)
- De milyen iterátoron fusson a szumma?

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit  
    return accu == num
```

```
def is_munchausen(num):  
    return sum(  
        ) == num
```

Münchausen-szám teszt (naiv, list comprehension)

- Végig megyünk a számjegyeken

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit  
    return accu == num
```

```
def is_munchausen(num):  
    return sum(  
        for ch in str(num)) == num
```

Münchausen-szám teszt (naiv, list comprehension)

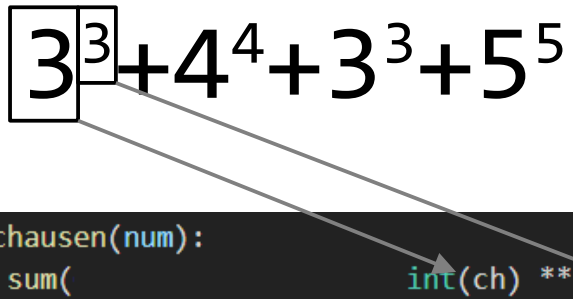
- És ugye vesszük a hatványokat

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit  
    return accu == num
```

```
def is_munchausen(num):  
    return sum(  
        int(ch) ** int(ch) for ch in str(num)) == num
```

Münchausen-szám teszt (naiv, list comprehension)

- És ugye vesszük a hatványokat

$$3^3 + 4^4 + 3^3 + 5^5$$
A diagram with two arrows originating from the first '3' in the equation. One arrow points to the first 'int(ch)' in the list comprehension, and the other points to the second 'int(ch)'.

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit  
    return accu == num
```

```
def is_munchausen(num):  
    return sum(  
        int(ch) ** int(ch) for ch in str(num)) == num
```


Münchausen-szám teszt (naiv, list comprehension)

- És ugye vesszük a hatványokat
- De mi ez a lyuk?!?!?!?!?!?!?!?

$$3^3 + 4^4 + 3^3 + 5^5$$

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit  
    return accu == num
```

```
def is_munchausen(num):  
    return sum(  
        int(ch) ** int(ch) for ch in str(num)) == num
```

Münchausen-szám teszt (naiv, list comprehension)

- Ez a csúnya hack megoldásunk mert $0^{**}0=1$, míg a feladatban $0^{**}0=0$!

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit  
    return accu == num
```

```
def is_munchausen(num):  
    return sum(0 if ch == "0" else int(ch) ** int(ch) for ch in str(num)) == num
```

Münchausen-szám teszt (naiv, list comprehension)

- Késznek látszik!

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += digit ** digit  
    return accu == num
```

```
def is_munchausen(num):  
    return sum(0 if ch == "0" else int(ch) ** int(ch) for ch in str(num)) == num
```

Münchausen-szám teszt (naiv, list comprehension)

- Teszteljük! (link:
https://github.com/rkeeves/munchausen/blob/main/src/e04_munchausen_property.py
)

```
def is_munchausen(num):  
    return sum(0 if ch == "0" else int(ch) ** int(ch) for ch in str(num)) == num
```

Münchausen-szám teszt (naiv, list comprehension)

- Teszteljük!

```
1 -> [0]
10 -> [0, 1]
100 -> [0, 1]
1000 -> [0, 1]
10000 -> [0, 1, 3435]
100000 -> [0, 1, 3435]
1000000 -> [0, 1, 3435]
```

```
def is_munchausen(num):
    return sum(0 if ch == "0" else int(ch) ** int(ch) for ch in str(num)) == num
```

Münchausen-szám teszt (naiv, list comprehension)

- Teszteljük!
- Jónak néz ki

```
1 -> [0]
10 -> [0, 1]
100 -> [0, 1]
1000 -> [0, 1]
10000 -> [0, 1, 3435]
100000 -> [0, 1, 3435]
1000000 -> [0, 1, 3435]
```

```
def is_munchausen(num):
    return sum(0 if ch == "0" else int(ch) ** int(ch) for ch in str(num)) == num
```

Münchausen-szám teszt (naiv, list comprehension)

- Egyébként ízlések és pofonok...

```
def is_munchausen(num):  
    accu = 0  
    for digit_char in str(num):  
        digit = int(digit_char)  
        accu += 0 if digit == 0 else digit ** digit  
    return accu == num
```

```
def list_munchausen(up_to):  
    accu = []  
    for num in range(0, up_to):  
        if is_munchausen(num):  
            accu.append(num)  
    return accu
```

```
def is_munchausen(num):  
    return sum(0 if ch == "0" else int(ch) ** int(ch) for ch in str(num)) == num  
  
def list_munchausen(up_to):  
    return [num for num in range(0, up_to) if is_munchausen(num)]
```

Münchausen-számok naiv összefoglaló

- Nagyon sok felesleges számítást végzünk

Münchausen-számok naiv összefoglaló

- Nagyon sok felesleges számítást végzünk
- Kit érdekel?

Münchausen-számok naiv összefoglaló

- Nagyon sok felesleges számítást végzünk
- Kit érdekel?
- Baloldalon a runtime sec-ban

```
[0.000000s] 1 -> [0]  
[0.000000s] 10 -> [0, 1]  
[0.000968s] 100 -> [0, 1]  
[0.001024s] 1000 -> [0, 1]  
[0.018958s] 10000 -> [0, 1, 3435]  
[0.225389s] 100000 -> [0, 1, 3435]  
[2.445194s] 1000000 -> [0, 1, 3435]
```

Münchausen-számok naiv összefoglaló

- Nagyon sok felesleges számítást végzünk
- Kit érdekel?
- Baloldalon a runtime sec-ban

```
[0.000000s] 1 -> [0]
[0.000000s] 10 -> [0, 1]
[0.000968s] 100 -> [0, 1]
[0.001024s] 1000 -> [0, 1]
[0.018958s] 10000 -> [0, 1, 3435]
[0.225389s] 100000 -> [0, 1, 3435]
[2.445194s] 1000000 -> [0, 1, 3435]
```

- Minden nagyságrendi lépéssel a futás idő is egy nagyságrendet lép minimum
- Mi lesz ha 10^9 -ig kell keresni? :(
- Be lehet-e gyorsítani?